ELSEVIER

Available at

www.ElsevierComputerScience.com

POWERED BY SCIENCE DIRECT®

Theoretical Computer Science 311 (2004) 199–220

Theoretical Computer Science

www.elsevier.com/locate/tcs

Limiting partial combinatory algebras☆

Yohji Akama

Mathematical Institute, Tohoku University, Aoba, Sendai, Miyagi 980-8578, Japan

Received 17 April 2002; received in revised form 11 March 2003; accepted 12 June 2003
Communicated by G.D. Plotkin

Abstract

From every partial combinatory algebra $\mathscr{A}$ we construct another partial combinatory algebra a-lim($\mathscr{A}$). In a-lim($\mathscr{A}$), every representable partial numerical function $\varphi(n)$ is exactly of the form $\lim_t \xi(t,n)$ for some representable partial numerical function $\xi(t,n)$ of $\mathscr{A}$. The partial combinatory algebra a-lim($\mathscr{A}$) is a quotient of $\mathscr{A}$ by a partial equivalence relation, and is equipped with a limit structure in the sense that each element of a-lim($\mathscr{A}$) is the limit of a countable sequence of $\mathscr{A}$-elements. In this paper, we discuss the limit structures for $\mathscr{A}$ in terms of Barendregt's range property (if the range of a combinator is finite, then it is a singleton). Moreover, we repeat the construction a-lim($-$) transfinite times to interpret infinitary λ-calculi. Finally, we attempt to interpret type-free $\lambda\mu$-calculus by introducing another partial applicative structure which has an asynchronous application operator that allows a parallel limit operation.

Keywords: Realizability interpretation; Partial equivalence relation; Limiting recursive; $\lambda\mu$-calculus

1. Introduction

Partial combinatory algebras (PCA) are partial applicative structures axiomatized by the same axioms as combinatory algebras, except that the application operators can be partial operators. The PCAs are important with regards to the realizability interpretations of intuitionistic logics. The realizability interpretations extract the computational content from intuitionistic logic's proofs as programs. Using PCAs to carry out the realizability interpretations, we can obtain 'realizability' models of typed term calculi and constructive set theories in which we can do mathematics; cf. tripos construction ([10]).

A new realizability interpretation is introduced by Nakata and Hayashi [13] to extract the computational content of semi-classical logic's proofs as approximation algorithms.

☆ The previous version appeared in the proceedings of CSL'01.

E-mail address: akama@math.tohoku.ac.jp (Y. Akama).

0304-3975/$ - see front matter
doi:10.1016/S0304-3975(03)00360-8

They consider Gold's limiting recursive functions [8], which are originally introduced to formulate the learning processes of machines. We say a partial function $\xi(t,\vec{x})$ *guesses* a partial function $\varphi(\vec{x})$, if

$$\varphi(\vec{x}) = y \;\Leftrightarrow\; \exists t_0 \forall t > t_0 . \xi(t,\vec{x}) = y \;\Leftrightarrow\; \lim_t \xi(t,\vec{x}) = y. \tag{1}$$

Then $\xi(t,\vec{x})$ is called a *guessing function* of $\varphi(\vec{x})$, or $\varphi(\vec{x})$ is the *limiting function* of $\xi(t,\vec{x})$. And t is called a *limit variable*. So, $\varphi(\vec{x})$ is approximated[1] by $\xi(t,\vec{x})$. They prove that the set of the partial functions guessed by partial recursive functions is a structure called *basic recursive function theory* (see Strong [17] and Wagner [18]). By using this structure, Nakata and Hayashi [13] note that some functions guessed by recursive functions are realizers of the following Σ^0_2-*double negation elimination axiom*:

$$\neg\neg\exists y \forall x . g(x,y) = 0 \to \exists y \forall x . g(x,y) = 0.$$

Also, they show impressive usage of the semi-classical principle for mathematics and for software synthesis.

If the set of operations guessed by an operation of a PCA $\mathscr{A}$ is again a PCA a-lim($\mathscr{A}$), then by carrying out Nakata–Hayashi realizability interpretation, using the PCA a-lim($\mathscr{A}$) instead, we can construct 'realizability' models of semi-classical formal theories of numbers and numerical functions. By using the limit structure of the PCA, we can interpret semi-classical typed term calculi (e.g., weak versions of typed Parigot's $\lambda\mu$-calculus, and typed term calculi with control operators).

Motivated by the above, we introduce another PCA a-lim($\mathscr{A}$), which is constructed from a given PCA $\mathscr{A}$. It is a quotient construction by a partial equivalence relation (PER), and our idea is the following:

- limit variable t is the clock of the processing element (MPU).
- the guessing function $\xi(-,x)$ is a generator of a stream $\langle \xi(0,x), \xi(1,x), \ldots \rangle$,
- the limiting function $\lim_t \xi(t,x)$ is the stream modulo a symmetric, transitive relation $\sim$.

  Two streams are related with the partial equivalence relation $\sim$ if for all natural numbers t, except finite numbers, the two tth elements of the two streams have the same value.

In the PCA a-lim($\mathscr{A}$) every allowable limit is exactly of the form, $\lim_t a_t$, such that the infinite sequence $\langle a_t \in \mathscr{A} \rangle_t$ is:

(**R**) indexed by $\mathbb{N}$; and

(**N**) generated inside the PCA $\mathscr{A}$.

We call this $\lim_t a_t$ an autonomous limit, and prove that every representable partial numerical function of a-lim($\mathscr{A}$) is exactly a limiting partial function guessed by a representable partial numerical function of $\mathscr{A}$ (see Section 3).

The more partial numerical functions are representable in a PCA, the more semi-classical principles can be interpreted by Nakata–Hayashi realizability interpretation

[1] The "guessed" comes from the computational learning theory.

over the PCA. Therefore, from any PCA we wish to construct a PCA which represents more partial numerical functions. We examine the conditions (**R**) and (**N**) of the PCA a-lim($\mathscr{A}$). Moreover, we consider the iteration of the constructions. If the indexing is not by $\mathbb{N}$ but by the PCA $\mathscr{A}$ itself, then the computation power does not necessarily increase due to Barendregt's range property (if the range of a combinator is finite, then it is a singleton). On the other hand, dropping the condition (**N**) results in a PCA n-lim($\mathscr{A}$), which we introduce in Section 4.2. The PCA n-lim($\mathscr{A}$) represents every partial numerical function. The ω-iterated application of a-lim to $\mathscr{A}$ produces a PCA a-lim$^{\omega}$($\mathscr{A}$) which represents every arithmetical partial function. Thus, both n-lim($\mathscr{A}$) and a-lim$^{\omega}$($\mathscr{A}$) provide Nakata–Hayashi realizability interpretation of the classical first-order arithmetic.

The PCA a-lim$^{\omega}$($\mathscr{A}$) may also be interesting, because it may interpret *infinitary* λ-*calculi* of Kennaway et al. [12] and Berarducci and Dezani [5].

In view of the relation between the limiting PCAs and the classical arithmetic, we consider the type-free $\lambda\mu$-calculus of Parigot [15]. The typed version corresponds to the classical logic. The typed (type-free resp.) version relates to a typed (type-free resp.) functional programming language with control operators, such as `call/cc`. Type-free $\lambda\mu$-calculus has μ-variables to represent continuation objects. By regarding μ-variables as infinite sequences of usual variables we can regard $\lambda\mu$-calculus as an infinitary λ-calculus.

To interpret a version of type-free $\lambda\mu$-calculus, we introduce another construction n-LIM($-$), which extends a given PCA $\mathscr{A}$ to a partial applicative structure n-LIM($\mathscr{A}$) such that:

- every allowed limit in n-LIM($\mathscr{A}$) is a parallel limit $\lim_{t_1,\ldots,t_k} f(t_1,\ldots,t_k,\vec{x})$,
- the application operator is asynchronous.

With the help of the concurrency theory, a branch of computer science, we try to clarify the parallelism hidden in the parallel limits of n-LIM($-$). For details see Section 5.

In the next section, for partial numerical functions, we consider the limit operation lim and another limit operation known in computational learning theory. In terms of the arithmetical hierarchy [14], we calibrate how much computation power the limit operations increase.

Throughout this paper the symbol "$\simeq$" means "if one-side is defined, then the other side is defined as having the same value", while the symbol "$=$" means that "both sides are defined as having the same value". The symbol "$\downarrow$" means "is defined". So, $t\downarrow$ is equivalent to an equation $t=t$. For any operation f, we assume that $f(a_1,\ldots,a_n)\downarrow$ implies $a_1\downarrow,\ldots$ and $a_n\downarrow$. The set of partial functions from A to B is denoted by $A\rightharpoonup B$. We write $\overset{\infty}{\forall} x\in A.\varphi(x)$ to express that "$\{x\in A\mid\neg\varphi(x)\}$ is a finite set".

A symmetric, transitive relation R on a non-empty set A is called a *partial equivalence relation*. So, R is an equivalence relation on a set $\{x\mid xRx\}$. The quotient set of A by R is $\{x\in A\mid xRx\}/R$. Any element of the set is of the form $[a]_R$ for some $a\in A$.

For any set X of natural numbers, the characteristic function is denoted by c_X, and let $\hat{c}_X(z)$ be the sequence number $\langle c_X(0),\ldots,c_X(z-1)\rangle$.

The set of partial recursive functions is denoted by **PRF**, and the set of natural numbers by $\mathbb{N}$. For all natural numbers e and x, the expression $\{e\}(x)$ is the eth unary partial numerical function applied to x.

2. Limiting recursion

Definition 1. For every class $\mathscr{F}$ of partial numerical functions, $\lim(\mathscr{F})$ denotes the set of partial numerical functions guessed by a partial numerical function of $\mathscr{F}$.

Nakata and Hayashi [13] showed the operation $\lim(-)$ has a good property. Here we review one of their results. We recall *ω-basic recursive function theory with a successor function* of Strong [17] and Wagner [18]. A class of partial numerical functions is called an *ω-basic recursive function theory with a successor function* (ω-BRFT with suc, for short), if it contains the following initial functions and is closed under the composition. The initial functions are the enumeration functions $\varphi_n(e,x_1,\ldots,x_n)$ for all $n\in\mathbb{N}$, the S_n^m-functions for all $m,n\in\mathbb{N}$, the successor, the constant functions, the projections, and the discriminator.

Theorem 2 (Nakata and Hayashi [13]). *If $\mathscr{F}$ is an ω-BRFT with suc, then the set* $\lim(\mathscr{F})$ *is also an ω-BRFT with suc.*

The set $\mathscr{F}$ is a subset of $\lim(\mathscr{F})$, because every partial function $\varphi(n_1,\ldots,n_k)\in\mathscr{F}$ is guessed by itself composed with projections, that is,

$$\varphi(\pi_2^{k+1}(t,n_1,\ldots,n_k),\ldots,\pi_{k+1}^{k+1}(t,n_1,\ldots,n_k)),$$

where each π_i^{k+1} returns the ith argument.

We calibrate the increase of the computational power by the construction $\lim(-)$. We assume the knowledge of arithmetical hierarchy and complete sets with respect to m-reducibility. The standard textbook is [14].

When considering sets of natural numbers, both the arithmetical hierarchy and the halting set $\mathscr{K}$ correspond well to the limit operations, in a sense of:

Proposition 3 (Shoenfield's limit lemma, Oddifreddi [14]). *For any set $A\subseteq\mathbb{N}$, the followings are equivalent*:
(1) *A belongs to a class Δ_2^0,*
(2) *A is recursive in the halting set $\mathscr{K}$,*
(3) *The characteristic function $c_A(x)$ is $\lim_t g(t,x)$ for some total recursive function g.*

When we take partial numerical functions into account, the increase of computation power by the limit operation lim is unexpectedly large.

Proposition 4 (Yamazaki [19], or folklore). *For every partial function, if it is guessed by a partial recursive function, then the graph and the domain both belong to a class Σ_3^0 of the arithmetical hierarchy. Moreover, there is a partial function f such that f is guessed by a partial recursive function and the graph of f and the domain of f are both complete with respect to the m-reducibility.*

Proof. Let $\mathscr{W}_n$ be the domain of the nth unary partial function $\{n\}(-)$. $\mathbf{Cof}=\{n\mid \mathscr{W}_n$ is cofinite$\}=\{n\mid \exists s\forall t>s.t\in\mathscr{W}_n\}$ is Σ_3^0-complete [14, Proposition X.9.11]. Define a

partial function by

$$\xi_{\mathbf{Cof}}(t,n) = \begin{cases} 1 & \text{if } t \in \mathscr{W}_n, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Then

$$x \in \mathbf{Cof} = \operatorname{dom}\left(\lim_t \xi_{\mathbf{Cof}}(t,n)\right) \;\Leftrightarrow\; (x,1) \in \operatorname{graph}\left(\lim_t \xi(t,n)\right). \qquad \square \tag{2}$$

This settles Gold's question. Gold [8] calls a set S *limiting partial r.e.* if there is a partial recursive function $\xi(t,x)$ such that

$$\lim_t \xi(t,x) = 1 \;\Leftrightarrow\; x \in S. \tag{3}$$

In his work [8, p. 32,l.6] he asks the exact location in the arithmetical hierarchy. We can answer by:

Corollary 5. *Every limiting partial r.e. set is exactly a* Σ_3^0*-set.*

Proof. By (1) and (3), we have the following. For every limiting partial r.e. set S, there exists a partial recursive function $\xi(t,x)$ such that for all x we have $x \in S \Leftrightarrow \exists t_0 \forall t > t_0.\ \xi(t,x) = 1$. Because the ternary relation $\xi(t,x) = y$ is a Σ_1^0-relation, the set S is a Σ_3^0-set.

Every Σ_3^0-set A is m-reduced to the set **Cof** by some recursive function f, because **Cof** is Σ_3^0-complete. By (2), we have

$$x \in A \;\Leftrightarrow\; f(x) \in \mathbf{Cof} \;\Leftrightarrow\; \lim_t \xi_{\mathbf{Cof}}(t, f(x)) \downarrow .$$

From the partial recursive function $\xi_{\mathbf{Cof}}(t,x)$ and the recursive function $f(x)$, define a partial recursive function $\zeta(t,x)$ by

$$\zeta(t,x) \simeq \begin{cases} 1 & \text{if } \xi_{\mathbf{Cof}}(t, f(x)) = \xi_{\mathbf{Cof}}(t+1, f(x)); \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Then,

$$x \in A \;\Leftrightarrow\; \lim_t \xi_{\mathbf{Cof}}(t, f(x)) \downarrow \;\Leftrightarrow\; \lim_t \zeta(t,x) = 1.$$

Therefore A is limiting partial r.e. $\square$

The following limit operation li, known in computational linguistics [11] is applied for Shoenfield's limit lemma, and for the following theorem about partial functions.

The limit operation li returns the "last" guess in a sense of the definition:

Definition 6. For every partial function $\xi(t,\vec{x})$, we define

$$\operatorname{li}_t \xi(t,\vec{x}) = y \;\Leftrightarrow\; \exists t_0(\xi(t_0,\vec{x}) = y \;\&\; \forall t > t_0 \forall z(\xi(t,\vec{x}) = z \Rightarrow z = y)).$$

For every class $\mathscr{F}$ of partial numerical functions, we define

$$\mathrm{li}(\mathscr{F}) = \{\mathrm{li}_t\, \xi(t,x) \mid \xi \in \mathscr{F}\}.$$

Theorem 7. *For any partial function* $\varphi(\vec{x})$, *the followings are equivalent*:
(1) φ *has* Σ_2^0*-graph*,
(2) φ *is partial recursive in the halting set* $\mathscr{K}$,
(3) $\varphi(\vec{x}) \simeq \mathrm{li}_t\, \xi(t,\vec{x})$ *for some partial recursive function* ξ.

Proof. The equivalence between (1) and (2) is derived from relativization of graph theorem and so on. See [14, Proposition II.1.11] or some textbooks on recursion theory.

First we prove that (3) implies (1). Every $\xi(t,\vec{x}) \in \mathbf{PRF}$ has a Σ_1^0-set as the graph. From the definition of the limit operation li_t, we can show that $\{(\vec{x}, y) \mid \mathrm{li}_t\, \xi(t,\vec{x}) = y\}$ is indeed a Σ_2^0-set. So, every partial function $\varphi \in \mathrm{li}(\mathbf{PRF})$ has Σ_2^0-graph.

Then we prove the implication from (2) to (3). Because φ is a partial recursive in the halting set $\mathscr{K}$, we can then apply *normal form theorem for partial recursive restricted functionals* [14]. Let n be the number of arguments of φ. Then there are a number e, a primitive recursive function U, and a primitive recursive predicate $T_{1,n}$ of only numerical variables, such that

$$\varphi(\vec{x}) \simeq \{e\}^{\mathscr{K}}(\vec{x}) \simeq U(\mu z.T_{1,n}(e,\vec{x},\hat{c}_{\mathscr{K}}(z),z)).$$

Recall that for all $e,\vec{x},u \in \mathbb{N}$ there exists at most one $z \in \mathbb{N}$ such that $T_{1,n}(e,\vec{x},u,z)$. For the Kleene's T predicate, we then write

$$\mathscr{K}_s = \{z \mid \exists t < s.Tzzt\}.$$

The characteristic function $c_{\mathscr{K}_s}(z)$ is a recursive function of s and z, because the ternary predicate T is primitive recursive. We now show that $\varphi(\vec{x}) \simeq \mathrm{li}_s\, \{e\}^{\mathscr{K}_s}(\vec{x})$. In the terminating oracle computation, oracle $c_{\mathscr{K}}$ is queried only finitely many times. So,

$$\varphi(\vec{x}) \downarrow \;\Leftrightarrow\; \exists z \overset{\infty}{\forall} s.T_{1,n}(e,x,c_{\mathscr{K}_s}(z),z).$$

From the construction of the predicate $T_{1,n}$, if $T_{1,n}(e,x,u,z_i)$ $(i=1,2)$, then $z_1 = z_2$. So, $\varphi(\vec{x}) \simeq \mathrm{li}_s\, \{e\}^{\mathscr{K}_s}(\vec{x})$. Because $\xi(s,\vec{x}) \simeq \{e\}^{\mathscr{K}_s}(\vec{x})$ is a partial recursive function of s and $\vec{x}$, this completes the proof. □

Corollary 8. *The set* $\mathrm{li}(\mathbf{PRF})$ *is a* BRFT *with suc.*

3. Autonomous limit

A *partial combinatory algebra* (PCA, *for short*) is $\mathscr{A} = \langle |\mathscr{A}|, \cdot, s, k\rangle$ such that $\cdot$ is a binary partial operator (*application operator*) on a set $|\mathscr{A}|$, and $s,k \in |A|$ are distinct elements subject to

$$(k \cdot a) \cdot b = a, \quad (s \cdot a) \cdot b \downarrow, \quad ((s \cdot a) \cdot b) \cdot c \simeq (a \cdot c) \cdot (b \cdot c).$$

As usual, we adopt the convention that the application operator $\cdot$ is left associative. Examples of PCA are the PCA $\mathscr{N}$ of the normal combinatory terms, $\langle \mathbb{N}, \{-\}(-)\rangle$, and the set of values of the call-by-value λ-calculus. $\mathscr{A}, \mathscr{B}, \ldots$ range over PCAs.

Given a PCA, we can simulate a λ-abstraction; for a "polynomial" $t[x]$ over a PCA, there is a "polynomial" $\lambda x.t[x]$ such that $(\lambda x.t[x]) \cdot a \simeq t[a]$. We have only to define $\lambda x.t[x]$ as follows:

$$\begin{aligned} \lambda x.a &= k \cdot a, \\ \lambda x.x &= (s \cdot k) \cdot k, \\ \lambda x.(t[x] \cdot t'[x]) &= s \cdot (\lambda x.t[x]) \cdot (\lambda x.t'[x]). \end{aligned}$$

The *Church numeral* of a natural number n, denoted by $\bar{n}$, is a polynomial $\lambda y \lambda x.y \cdot (\cdots (y \cdot x) \cdots)$ with the y successively applied n-times to x.

We say a partial numerical function $\psi(t, n_1, \ldots, n_k)$ is *represented* by an element $a \in \mathscr{A}$, if $(\cdots ((a \cdot \bar{t}) \cdot \overline{n_1}) \cdots) \cdot \overline{n_k} = \bar{m}$ is equivalent to $\psi(t, n_1, \ldots, n_k) = m$. The set of partial numerical functions representable in $\mathscr{A}$ is denoted by RpFn($\mathscr{A}$). It is well known that RpFn($\mathscr{A}$) contains **PRF**.

Given a PCA $\mathscr{A}$, we construct a PCA a-lim($\mathscr{A}$) such that RpFn(a-lim($\mathscr{A}$))= lim(RpFn($\mathscr{A}$)).

Definition 9. Given a PCA $\mathscr{A}$, define a partial equivalence relation over the set $|\mathscr{A}|$ as follows:

$$a \sim b : \Leftrightarrow \overset{\infty}{\forall}\, t \in \mathbb{N}.\ (a \cdot \bar{t} = b \cdot \bar{t}).$$

So, $a \sim a$ is equivalent to "$a \cdot \bar{t}$ is defined but in finitely many cases". The PCA $\mathscr{N}$ has an element a such that $a \sim a$ is false. Let i be skk and Δ be sii. Then Δ applied to itself is undefined in $\mathscr{N}$. When $s(k\Delta)(k\Delta) \in \mathscr{N}$ is applied to any Church numerals, it becomes undefined.

Definition 10 (Autonomous limit of PCA). Let $\mathscr{A} = \langle |\mathscr{A}|,\ \cdot,\ s,\ k\rangle$ be any PCA. The extension a-lim($\mathscr{A}$) of $\mathscr{A}$ by the autonomous limits is a PCA a-lim($\mathscr{A}$) $= \langle$ |a-lim($\mathscr{A}$)|, $*, \mathbf{s}, \mathbf{k}\rangle$, where:

- |a-lim($\mathscr{A}$)| is a quotient set $\{a \in |\mathscr{A}| \mid a \sim a\}/\sim$,
- $\mathbf{s} = [k \cdot s]_\sim$, $\mathbf{k} = [k \cdot k]_\sim$ ("s, k for any value t", where t is "time"),
- $[a]_\sim * [b]_\sim = [(s \cdot a) \cdot b]_\sim$ ("synchronous application").

The operation $*$ is well defined. Suppose $a \sim a', b \sim b'$ and $s \cdot a \cdot b \cdot \bar{n} \simeq (a \cdot \bar{n}) \cdot (b \cdot \bar{n})$ is defined for values of n which are large enough. Then, because $a \cdot \bar{n} = a' \cdot \bar{n}$ and $b \cdot \bar{n} = b' \cdot \bar{n}$, we have $s \cdot a \cdot b \cdot \bar{n} \simeq (a \cdot \bar{n}) \cdot (b \cdot \bar{n}) \simeq (a' \cdot \bar{n}) \cdot (b' \cdot \bar{n}) \simeq s \cdot a' \cdot b' \cdot \bar{n}$ for n large enough. Therefore, $[a] * [b] \simeq [a'] * [b']$.

Theorem 11. *If $\mathscr{A}$ is a* PCA, *then the structure* a-lim($\mathscr{A}$) *is a* PCA *such that* RpFn (a-lim($\mathscr{A}$)) = lim(RpFn($\mathscr{A}$)).

Proof. In order to prove that a-lim($\mathscr{A}$) is a PCA, we assume $t \in \mathbb{N}$ be sufficiently large:

- $\mathbf{s} * [a] * [b] * [c] \simeq [s(s(s(ks)a)b)c]$ while $[a] * [c] * ([b] * [c]) \simeq [s(sac)(sbc)]$.

$$\begin{aligned} & s(s(s(ks)a)b)c\bar{t} \\ \simeq\ & (ks\bar{t})(a\bar{t})(b\bar{t})(c\bar{t}) && \text{by an axiom for } s \\ \simeq\ & s(a\bar{t})(b\bar{t})(c\bar{t}) && \text{by using the axiom for } k \text{ with } \bar{t}\downarrow \\ \simeq\ & s(sac)(sbc)\bar{t} && \text{by an axiom for } s. \end{aligned}$$

So $s(s(s(ks)a)b)c \sim s(sac)(sbc)$. Therefore, $\mathbf{s} * [a] * [b] * [c] \simeq [a] * [c] * ([b] * [c])$.

- $\mathbf{s} * [a] * [b] \simeq [s(s(ks)a)b]$.

$$\begin{aligned} s(s(ks)a)b\bar{t} &\simeq ks\bar{t}(a\bar{t})(b\bar{t}) && \text{by an axiom of } s \\ &\simeq s(a\bar{t})(b\bar{t}) && \text{by using the axiom of } k \text{ with } \bar{t}\downarrow. \end{aligned}$$

It is always defined because of $a\bar{t}\downarrow$, $b\bar{t}\downarrow$ and an axiom of s. So $\mathbf{s} * [a] * [b]\downarrow$.

- $\mathbf{k} * [a] * [b] = [s(s(kk)a)b]$ while

$$\begin{aligned} s(s(kk)a)b\bar{t} &\simeq (kk\bar{t})(a\bar{t})(b\bar{t}) \\ &\simeq k(a\bar{t})(b\bar{t}) && \text{by using the axiom of } k \text{ with } \bar{t}\downarrow \\ &\simeq a\bar{t} && \text{by } b\bar{t}\downarrow. \end{aligned}$$

So $s(s(kk)a)b \sim a$. Thus $\mathbf{k} * [a] * [b] = [a]$.

Therefore, a-lim($\mathscr{A}$) is indeed a PCA.

Next we prove that RpFn(a-lim($\mathscr{A}$)) = lim(RpFn($\mathscr{A}$)). Both RpFn(a-lim($\mathscr{A}$)) and lim(RpFn($\mathscr{A}$)) have the partial functions which domains are empty. Let φ be a unary partial function which domain is not empty. Then

$$\begin{aligned} & \varphi \in \mathrm{RpFn}(\text{a-lim}(\mathscr{A})) \\ \Leftrightarrow\ & \exists a \in \mathscr{A}(a \sim a\ \&\ \forall n, m([a]_\sim * \bar{\mathbf{n}} = \bar{\mathbf{m}} \text{ iff } \varphi(n) = m)). \end{aligned} \tag{4}$$

We note that the nth Church numeral $\bar{\mathbf{n}}$ of a-lim($\mathscr{A}$) is the equivalence class of the map constantly equal to the nth Church numeral $\bar{n}$ of $\mathscr{A}$, or the equivalence class $[k\bar{n}]_\sim$. So, (4) is equivalent to the following:

$$\exists a \in \mathscr{A}\left(\overset{\infty}{\forall}\ t.a\bar{t}\downarrow\ \ \&\ \forall n, m\left(\left(\overset{\infty}{\forall}\ t \in \mathbb{N}.a\bar{t}\bar{n} = \bar{m}\right)\ \text{iff}\ \varphi(n) = m\right)\right). \tag{5}$$

Let ξ be a partial function represented by a. Then (5) implies

$$\forall n, m\left(\lim_t \xi(t, n) = m\ \text{iff}\ \varphi(n) = m\right). \tag{6}$$

We establish that (6) implies (5) below. As the domain of φ is not empty, we have $\varphi(l)\downarrow$ for some l. Therefore, if (6) holds, then $\overset{\infty}{\forall}\ t.\xi(t, l)\downarrow$. So, every $a \in \mathscr{A}$ representing ξ satisfies $\overset{\infty}{\forall}\ t.a\bar{t}\downarrow$. Thus, we have (5) $\Leftrightarrow$ (6).

Hence,

$$\varphi \in \mathrm{RpFn}(\text{a-lim}(\mathscr{A}))\ \Leftrightarrow\ (6)\ \Leftrightarrow\ \varphi \in \lim(\mathrm{RpFn}(\mathscr{A})).$$

For any φ with the arity being greater than 1, we can similarly prove $\varphi \in \mathrm{RpFn}(\text{a-lim}(\mathscr{A}))$ if and only if $\varphi \in \lim(\mathrm{RpFn}(\mathscr{A}))$. □

Following Grätzer [9], we define homomorphisms between PCAs.

Definition 12 (homomorphism). A function f from a PCA $\mathscr{A}$ to a PCA $\mathscr{B}$ is a homomorphism, if f preserves the operators as relations. i.e.,
- $f(s)=\mathbf{s}$,
- $f(k)=\mathbf{k}$,
- $ab=c$ implies $f(a)f(b)=f(c)$.

An injective, surjective homomorphism is called an isomorphism.

Example 13. For every model $(\mathscr{D},\cdot,[\![-]\!])$ of the combinatory logic, $(\mathscr{D},\cdot)$ is a PCA and $[\![-]\!]$ becomes a homomorphism from the PCA $\mathscr{N}$ to the PCA $\mathscr{D}$. Even if $a\cdot b$ is not defined in $\mathscr{N}$, $[\![a]\!]\cdot[\![b]\!]$ is always defined.

Definition 14 (Canonical injective homomorphism). Define the function $\iota_{\mathscr{A}}:\mathscr{A}\to$ a-lim($\mathscr{A}$) by $\iota_{\mathscr{A}}(a)=[k\cdot a]_{\sim}$. Note that $\iota_{\mathscr{A}}(a)$ is the equivalence class of the map constantly equal to a.

Theorem 15. *For the function $\iota_{\mathscr{A}}$ defined above*:
(1) *$\iota_{\mathscr{A}}$ is an injective, non-surjective homomorphism.*
(2) $[a]_{\sim}=\iota_{\mathscr{A}}(b)\Leftrightarrow \overset{\infty}{\forall}\, t\in\mathbb{N}.\ a\cdot\bar{t}=b$ $(\Leftrightarrow$ "$\lim_t a\cdot\bar{t}=b$").
(3) $a\cdot a'=b\Leftrightarrow \iota_{\mathscr{A}}(a)*\iota_{\mathscr{A}}(a')=\iota_{\mathscr{A}}(b)$.

Proof. The function $\iota_{\mathscr{A}}$ is clearly an injective homomorphism. Assume $\iota_{\mathscr{A}}$ is surjective. Then, $\mathscr{A}$ has an element a such that $a\cdot\bar{t}=\overline{t \bmod 2}$. Then we have $\exists b\overset{\infty}{\forall}\, t\in\mathbb{N}\,(a\cdot\bar{t}=b)$, which is a contradiction. The other claims are clear. □

Remark 16 (Undefinedness vs. divergence). In our PCA a-lim($\mathscr{A}$), undefinedness and divergence have separate definitions which should be clarified.

An element $[a]_{\sim}$ of our limiting PCA a-lim($\mathscr{A}$) is divergent, if:
(1) there is no $b\in\mathscr{A}$ such that $\overset{\infty}{\forall}\, n\in\mathbb{N}.a\bar{n}=b$, and
(2) $\overset{\infty}{\forall}\, n\in\mathbb{N}.a\bar{n}$ is defined in $\mathscr{A}$.

On the other hand, $[a]_{\sim}$ is undefined (i.e., $[a]_{\sim}\notin$ a-lim($\mathscr{A}$)), if condition (2) fails.

To give examples of divergence and undefinedness in our limiting PCAs, let us see the case where the PCA $\mathscr{A}$ is the PCA $\mathscr{N}$. In the a-lim($\mathscr{A}$), an element $[(SK)K]_{\sim}$ is divergent because it is the limit of the sequence

$$0,1,2,3,\ldots\ .$$

On the other hand, a-lim($\mathscr{N}$) has an element a such that a applied to itself is undefined. To see it, write I for $(SK)K$ and take $a=[K((SI)I)]_{\sim}$. A combinatory term $(SI)I$ belongs to the PCA $\mathscr{N}$. The self-application has no normal form, and is undefined in $\mathscr{N}$.

4. Constructing PCAs representing more functions

Each element of the PCA a-lim($\mathscr{A}$) is of the form $\lim_t a_t$, where:
(1) the parameter t can be any natural number,
(2) the sequence $\langle a_t\rangle_t$ is of the form $\langle a\cdot\bar{t}\rangle_t$ for some $a\in\mathscr{A}$. In this case, the sequence is "autonomously tracked" by an $\mathscr{A}$-element a.

To justify the necessity of the above two conditions, we discuss following alternative $\lim_t a_t$ for $\mathscr{A}$:
(**R**) the limit variable t of $\lim_t a_t$ is any element of $\mathscr{A}$ (see Section 4.1), or
(**N**) the sequence $\langle a_t\rangle_t$ is any countable sequence (see Section 4.2).

4.1. Range property and limit

We consider the following semantics of lim operation in a PCA $\mathscr{A}$:

$$\lim_a \xi(\vec{x},a) = y \quad\Leftrightarrow\quad \overset{\infty}{\forall}\, a\in|\mathscr{A}|.\, y=\xi(\vec{x},a). \tag{7}$$

This limit operation is useful when $\mathscr{A}$ is a PCA $\mathbb{N}$, but not when $\mathscr{A}$ is the PCA $\mathscr{M}$ of closed λ-terms modulo β-equality. Even from $\mathscr{M}$ we could construct another PCA $A(\mathscr{M})$ with the limit operation given in (7), the representable partial numerical functions

$$\mathrm{RpFn}(A(\mathscr{M})) = \mathrm{RpFn}(\mathscr{M})$$

by the range property studied by Barendregt [2, p. 517], namely, the range of any combinator is either a singleton or an infinite set.

Indeed, for any $f,x\in\mathscr{M}$, if $\lim_a(fxa)$ is convergent in the above sense, then the range $\{fxa\,|\,a\in\mathscr{M}\}$ of a combinator fx should be finite. Actually, it should be a singleton because of the range property. Therefore, the lim given in (7) is useless.

4.2. Limit along all the countable sequences

Definition 17. For every PCA $\mathscr{A}$, define a partial equivalence relation $\sim$ over the set $\mathbb{N}\rightharpoonup|\mathscr{A}|$ of partial functions from $\mathbb{N}$ to the set $|\mathscr{A}|$, as follows:

$$f\sim g:\ \Leftrightarrow\ \overset{\infty}{\forall}\, t\in\mathbb{N}.(f(t)=g(t)).$$

Definition 18 (Non-constructive limit of PCA). Let $\mathscr{A}=\langle|\mathscr{A}|,\cdot,s,k\rangle$ be any PCA. The extension n-lim($\mathscr{A}$) of $\mathscr{A}$ by the non-constructive limits, is a PCA n-lim($\mathscr{A}$) $=\langle|\text{n-lim}(\mathscr{A})|,*,\mathbf{s},\mathbf{k}\rangle$, where:
- $|\text{n-lim}(\mathscr{A})|$ is a quotient set $\{f:\mathbb{N}\rightharpoonup|\mathscr{A}|\,|\,f\sim f\}/\sim$,
- $\mathbf{s}=[x\mapsto s]_\sim$, $\mathbf{k}=[x\mapsto k]_\sim$ ("s,k for any value t", where t is "time"),
- $[f]_\sim * [g]_\sim = [x\mapsto f(x)\cdot g(x)]_\sim$ ("synchronous application").

Theorem 19. *If $\mathscr{A}$ is a* PCA, n-lim($\mathscr{A}$) *is a* PCA *such that* RpFn(n-lim($\mathscr{A}$)) *is the set of all partial numerical functions.*

Proof. Here, we only prove the second part. Since any partial numerical function of finite domain is representable in any PCA, every partial numerical m-ary function φ is represented by $[f]_\sim$ such that each $f(t) \in \mathscr{A}$ represents a partial numerical function $\varphi \upharpoonright \{0,1,\ldots,t-1,t\}^m$ of finite domain. □

We can define the canonical injective homomorphism of n-lim($\mathscr{A}$) and prove a theorem similar to Theorem 15. The construction n-lim(−) also preserves the partial algebras. By a partial algebra we mean it is axiomatized by possibly sorted partial operators subject to some equations by $=$ and $\simeq$.

We conjecture that the PCA a-lim($\mathscr{A}$) is the PCA n-lim($\mathscr{A}$) constructed inside the realizability topos $\mathsf{RT}(\mathscr{A})$. Cf. [10].

4.3. Repeating limits

In functional programming, we use infinite lists as streams (for input/output) and non-terminating recursive calls. These objects are usually unwinding or "limit" of finite objects. To analyze such infinite objects of interest, *infinitary* λ-calculi are introduced by Kennaway et al. [12] and Berarducci and Dezani [5]. Both calculi admit a term like $\lambda x.y^\omega x$, $\lambda x.y^\omega(x^\omega)$, and have terms with the limit operation $(-)^\omega$ being nested. So to interpret all of the infinitary λ-terms, it is necessary to repeat the limit constructions ω-times.

By using the Nakata–Hayashi realizability interpretation over a-lim$^\alpha(\mathscr{A})$, we want to obtain hierarchical view of semi-classical principles.

Definition 20 (α-times repeated limits). For every PCA $\mathscr{A}$, every ordinal number α, we define a PCA a-lim$^\alpha(\mathscr{A})$ and the canonical injective homomorphism $\iota^\alpha_\beta : \text{a-lim}^\beta(\mathscr{A}) \to \text{a-lim}^\alpha(\mathscr{A})$ for each ordinal number $\beta \leqslant \alpha$ such that:

- a-lim$^0(\mathscr{A}) = \mathscr{A}$, and ι^β_β is the identity;
- a-lim$^{\beta+1}(\mathscr{A}) = \text{a-lim}(\text{a-lim}^\beta(\mathscr{A}))$, and $\iota^{\beta+1}_\gamma = \iota_{\text{a-lim}^\beta(\mathscr{A})} \circ \iota^\beta_\gamma$;
- when α is a limit ordinal number, a-lim$^\alpha(\mathscr{A})$ is an inductive limit of

$$\langle \iota^\delta_\gamma : \text{a-lim}^\gamma(\mathscr{A}) \to \text{a-lim}^\delta(\mathscr{A}) \rangle_{0 \leqslant \gamma < \delta < \alpha},$$

and each ι^α_δ is a natural injection of the inductive limit.

We can similarly define n-lim$^\alpha(\mathscr{A})$.

Theorem 21. *For each ordinal number* $\alpha > 0$:

(1) *the set* a-lim$^\alpha(\mathscr{A})$ *is indeed a* PCA, *and the function* ι^α_β *is indeed an injective, non-surjective homomorphism for* $\beta < \alpha$; and

$$\text{(2)} \qquad \text{RpFn}(\text{a-lim}^\alpha(\mathscr{A})) = \begin{cases} \lim^\alpha(\text{RpFn}(\mathscr{A})) & \text{if } \alpha \text{ is finite,} \\ \bigcup_{\beta \in \mathbb{N}} \lim^\beta(\text{RpFn}(\mathscr{A})) & \text{otherwise.} \end{cases}$$

Proof. The first claim is proven by transfinite induction on α. The second claim, for the case α being finite, is proven by Theorem 11. For the case $\alpha = \omega + 1$, it is sufficient to note the following: any autonomous sequence $\langle a \cdot \vec{t} \rangle_{t \in \mathbb{N}}$ indexed over some

$a \in \text{a-lim}^{\omega}(\mathscr{A})$ is in fact indexed over some $a \in \text{a-lim}^{n}(\mathscr{A})$. Thus, its limit of the sequence belongs to $\text{a-lim}^{n+1}(\mathscr{A})$. Therefore, $\text{RpFn}(\lim^{\omega+1}(\mathscr{A})) \subseteq \text{RpFn}(\lim^{\omega}(\mathscr{A}))$. The proof of the opposite inclusion is trivial. In this way, we can prove the second claim for any infinite ordinal number α. □

5. An interpretation of type-free $\lambda\mu$-calculus

Parigot [15] introduced the *typed* $\lambda\mu$*-calculus* which corresponds to classical propositional logic via Curry–Howard isomorphism. By forgetting the types in the $\lambda\mu$-calculus, we obtain a *type-free* $\lambda\mu$*-calculus.* Both calculi are related to typed and type-free programming languages with control operators respectively (e.g., `call/cc`, $\mathscr{C}$ introduced by Felleisen and Friedman [6], exception, etc.).

Since Nakata and Hayashi [13] interpreted weak classical logic by a limiting realizability interpretation, it is natural to ask whether we can interpret a $\lambda\mu$-calculus by a λ-model which has a limit structure. We concentrate on the type-free version of $\lambda\mu$-calculus.

Type-free $\lambda\mu$-calculus is specified by defining $\lambda\mu$-terms and the reduction rules (the β-reduction rule and the μ-reduction rule). We give the definition below.

$\lambda\mu$-terms are generated by

$$M ::= c \mid x \mid MM \mid \lambda x.M \mid [\alpha]M \mid \mu\alpha.M.$$

An occurrence of α in $\mu\alpha.\cdots$ is called a *bound* occurrence of α. An occurrence of α which is not bound is called *free*. A $\lambda\mu$-term $[\alpha]M$ is regarded as application of M to the continuation bound to α. A $\lambda\mu$-term $\mu\alpha.M$ is regarded as functional abstraction over the continuation variable α on the level of continuation semantics. Here, $x, y, z, \ldots$ range over term-variables. And $\alpha, \beta, \gamma, \ldots$ range over μ*-variables* which are distinguished from the term-variables.

Now we recall mixed substitution. A *context* is generated by the above grammar with a special constant ( ). By replacing the occurrences of ( ) of a context $C(\,)$ with a $\lambda\mu$-term M, we obtain a $\lambda\mu$-term $C(M)$.

For any context $C(\,)$ and $\lambda\mu$-term N, we define a *mixed substitution* $\vartheta = [\,[\alpha] := C(\,)]$ as follows:

If N is a $\lambda\mu$-term, then $N\vartheta$ is also a $\lambda\mu$-term. If N does not have a free μ-variable α, then $N\vartheta$ is N. So, if N is a term-variable or $\mu\alpha.M$, then $N\vartheta \equiv N$. The mixed substitution commutes with a λ-abstraction and an application. The mixed substitution $\vartheta = [\,[\alpha] := C(\,)]$ satisfies the identities $([\beta]M)\vartheta \equiv [\beta](M\vartheta)$ and $(\mu\beta.M)\vartheta \equiv \mu\beta.(M\vartheta)$, provided $\beta \neq \alpha$. Finally, $([\alpha]M)[\,[\alpha] := C(\,)] \equiv C(M[\,[\alpha] := C(\,)])$.

The μ-reduction of the type-free $\lambda\mu$-calculus is specified by the following rule:

$$(\mu\alpha.M)N \to_{\mu} \mu\alpha.(M[\,[\alpha] := [\alpha]((\,)N)]).$$

In graphical presentation, the rule is

$$(\mu\alpha.(\cdots([\alpha]P)\cdots))N \to_{\mu} \mu\alpha.(\cdots([\alpha](P'N))\cdots). \tag{8}$$

An example of the μ-reduction is $(\mu\alpha.[\alpha](y[\alpha]x))z \to_{\mu} \mu\alpha.[\alpha](y([\alpha](xz))z)$.

5.1. Informal semantics of $\lambda\mu$

To introduce the semantics of $\lambda\mu$-calculus, based on streams, we first informally explain the idea.

5.1.1. μ-application(abstraction,reduction) = infinite applications(abstractions, β-reduction)

Consider an *informal* translation from the type-free $\lambda\mu$-calculus to a type-free calculus of infinitely long λ-terms. The translation rules are

$$[\alpha]P \mapsto \tilde{P}\alpha_0 \dots \alpha_m \dots \quad \text{and} \quad \mu\alpha.M \mapsto \lambda\alpha_0 \dots \alpha_m \dots .\tilde{M}. \tag{9}$$

Then, the above reduction rule (8) is translated to the next reduction rule

$$\begin{aligned} &(\lambda\alpha_0\alpha_1 \dots (\cdots(\tilde{P}\alpha_0\alpha_1 \dots)\cdots))\tilde{N} \\ &\to_\beta \lambda\alpha_0\alpha_1 \dots (\cdots(\tilde{P}'\tilde{N}\alpha_0\alpha_1 \dots)\cdots), \end{aligned} \tag{10}$$

which becomes a β-reduction between two infinitely long terms, if we rename bound variables to have $\lambda\alpha_1 \dots . (\cdots(\tilde{P}'\tilde{N}\alpha_1 \dots)\cdots) \equiv \lambda\alpha_0\alpha_1 \dots (\cdots(\tilde{P}'\tilde{N}\alpha_0\alpha_1 \dots)\cdots)$. The η-like reduction rule on continuation

$$\mu\alpha.[\alpha]P \to P \text{ (if } \alpha \text{ is not free in } P)$$

becomes just an *infinite* η-reduction (see Theorem 36).

This idea of Parigot [15] has been studied by Fujita [7] for the case of the typed $\lambda\mu$-calculus. In that particular case, the translation above precisely corresponds to Gödel translation and the length of $\alpha_0, \alpha_1, \dots$ is finite.

5.1.2. μ-variable = infinite stream

The idea of the translation rules (9) leads us to interpret

$$[\alpha]P \simeq \lim_t \tilde{P}\alpha_0\alpha_1 \dots \alpha_t \simeq [t \mapsto \tilde{P}\alpha_0\alpha_1 \dots \alpha_t]_\sim$$

where

$$\alpha_i \simeq \mathbf{car} * \overbrace{(\mathbf{cdr} * \cdots (\mathbf{cdr}}^{i} * \alpha)\cdots).$$

If we allow more general *continuation terms* than mere μ-variables, namely pure λ-terms stacked to a μ-variable,

$$\mathbf{cons} * M_1 * (\mathbf{cons} * M_2 * \cdots * (\mathbf{cons} * M_n * \alpha)\dots),$$

then we have the following *Swap rule* of Streicher–Reus's version of type-free $\lambda\mu$-calculus [16]:

$$[\mathbf{cons} * M * N]P \simeq [N](P * M).$$

Before we consider the interpretation of μ-abstraction, we try to make (10) rigorous.

5.1.3. μ-reduction causes delay in a stream

The translation result of the reduction rule (8) is $f \to g$ such that:

$$f(t) \simeq (\lambda a_0 \ldots a_t.(\cdots(\tilde{P}a_0 \ldots a_t)\cdots))\tilde{N} \simeq \lambda a_1 \ldots a_t.(\cdots(\tilde{P}\tilde{N}a_1 \ldots a_t)\cdots),$$
$$g(t) \simeq \lambda a_0 \ldots a_t.(\cdots(\tilde{P}\tilde{N}a_0 \ldots a_t)\cdots).$$

But, $f(t+1) \simeq g(t)$. Because it takes 1 'clock time' to compute g from f by β-reduction, a delay occurs as the extra computational time is required. Anyway, $[f]_\sim \neq [g]_\sim$ in the PCA n-lim($\mathscr{A}$). So, n-lim($\mathscr{A}$) does not interpret the calculus. Neither does a-lim($\mathscr{A}$).

To equate f and g given above, let us replace the symmetric, transitive relation $\sim$ with the smallest symmetric, transitive relation $\approx$ containing $\sim$ and the 'delay' rule $f \approx (t \mapsto f(t+1))$. Unfortunately, a quotient set $(\mathbb{N} \rightharpoonup |\mathscr{A}|)/\approx$ cannot have the synchronous application operator $[f]_\approx * [g]_\approx \simeq [t \mapsto f(t) \cdot g(t)]_\approx$ well defined.

5.2. Asynchronous applicative structure and parallel limit

Given a PCA $\mathscr{A}$, we introduce another partial algebra n-LIM($\mathscr{A}$) where an appropriate application operator can be defined.

The carrier set of n-LIM($\mathscr{A}$) is

$$\left\{ f \;\middle|\; \begin{array}{l} \exists n \geqslant 0.f : \mathbb{N}^n \rightharpoonup |\mathscr{A}| \\ \text{and } \exists t_0 \forall t_1 > t_0 \ldots \forall t_n > t_0.f(t_1, \ldots, t_n) \downarrow \end{array} \right\} \Big/ \sim,$$

where $\sim$ is an equivalence relation over $\bigcup_{n \geqslant 0}(\mathbb{N}^n \rightharpoonup |\mathscr{A}|)$, generated by the following two rules:

(1) The 'delay' rule.

$$(\mathbb{N}^n \xrightarrow{f} |\mathscr{A}|) \sim (\mathbb{N}^n \xrightarrow{id \times \cdots \times id \times suc \times id \cdots \times id} \mathbb{N}^n \xrightarrow{f} |\mathscr{A}|),$$

where *id* is the identity function on $\mathbb{N}$ and *suc* is the successor function.

As for $\times$, it is an associative operator and for all $f_i : A_i \rightharpoonup B_i$ we have $f_1 \times f_2 : A_1 \times A_2 \rightharpoonup B_1 \times B_2$ such that $(f_1 \times f_2)(a_1, a_2)$ is $(f_1(a_1), f_2(a_2))$ if each $f_i(a_i)$ is defined, and is undefined otherwise.

The rule is necessary to have $\lim_t f(t) \simeq \lim_t f(t+1)$.

(2) The 'exchange'-'weakening' rule.

$$(\mathbb{N}^n \xrightarrow{f} |\mathscr{A}|) \sim (\mathbb{N}^m \xrightarrow{(\pi^m_{\sigma(1)}, \pi^m_{\sigma(2)}, \ldots, \pi^m_{\sigma(n)})} \mathbb{N}^n \xrightarrow{f} |\mathscr{A}|),$$

where $m \geqslant n$, σ is a permutation on $\{1, \ldots, n\}$, and for each $k = 1, \ldots, m$ the function π^m_k returns the kth argument.

For all $f_i : A \rightharpoonup B_i$ we have $(f_1, \ldots, f_n) : A \rightharpoonup B_1 \times \cdots \times B_n$ such that $(f_1, \ldots, f_n)(a)$ is $(f_1(a), \ldots, f_n(a))$ if each $f_i(a)$ is defined, and it is undefined otherwise.

Remark 22 (Parallel limit vs. sequential limit). For any partial function $f : \mathbb{N}^k \rightharpoonup |\mathscr{A}|$ the *parallel limit* $\lim_{t_1, \ldots, t_k} f(t_1, \ldots, t_k)$ corresponds to $[f]_\sim \in$ n-LIM($\mathscr{A}$).

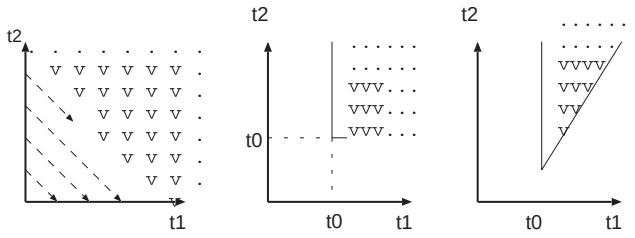

Fig. 1. The values of $f(t_1,t_2)$.

Specifically,

$$\begin{aligned} v &= \lim_{t_1,\ldots,t_k} f(t_1,\ldots,t_k) \\ \Leftrightarrow\ & \exists t_0\, \forall t_1 > t_0 \ldots \forall t_k > t_0 . v = f(t_1,\ldots,t_k). \end{aligned}$$

So, the parallel limit $\lim_{t_1,\ldots,t_k}$ is exactly the sequential limit $\lim_x$ where $x \in \mathbb{N}^k$ and $\mathbb{N}^k$ is ordered componentwise.

The parallel limit operation $\lim_{t_1,\ldots,t_k}$ is not definable by the sequential limit $\lim_t$ with $t \in \mathbb{N}$. But a parallel limit operation and the sequential limit operation are related as follows:

$$v = \lim_t f(j_0(t), j_1(t)), \tag{11}$$

$$\Longrightarrow v = \lim_{t_1,t_2} f(t_1,t_2), \tag{12}$$

$$\Longrightarrow v = \lim_{t_1} \lim_{t_2} f(t_1,t_2). \tag{13}$$

Here the function $j(m,n) = (m+n)^2+m$ is a numerical pairing, the function j_0 is the left-projection, and the function j_1 is the right-projection. All of j, j_0, j_1 are recursive. Note that if the values of $f(t_1,t_2)$ is as in Fig. 1, then we have (11),(12),(13), accordingly. So, $(11) \not\Leftarrow (12) \not\Leftarrow (13)$.

As is common, the contraction rule is interpreted as a communication (synchronization) of processes.[2] In defining the equivalence relation $\sim$, we cannot replace the 'exchange'-'weakening' rule with the 'exchange'-'weakening'-'contraction' rule: for $m \geqslant n$ and σ is a function on $\{1,\ldots,n\}$,

$$(\mathbb{N}^n \xrightarrow{f} |\mathscr{A}|) \sim (\mathbb{N}^m \xrightarrow{(\pi^m_{\sigma(1)},\pi^m_{\sigma(2)},\ldots,\pi^m_{\sigma(n)})} \mathbb{N}^n \xrightarrow{f} |\mathscr{A}|).$$

[2] A total differentiable function $f(x,y)$ of analysis and an analytic function $F(z)$ of complex variable $z = x + \mathrm{i}y$ allow us to calculate the derivatives f' and F' by either an iterated sequential limit $\lim_x \lim_y \ldots$ or a parallel limit $\lim_{x,y} \ldots$, or $\lim_z \ldots$. The variables x and y are cooperating and 'communicate' with each other in the calculation of the derivative. If x, y can be seen as clocks, f, F as a process, and f', F' as the generator of the process, then the total differentiability and being analytic can be seen as a cooperating and/or communicating process.

Definition 23 (An asynchronous application operator). For $f:\mathbb{N}^n \rightharpoonup |\mathscr{A}|$ and $g:\mathbb{N}^m \rightharpoonup |\mathscr{A}|$, define $[f]_\sim * [g]_\sim \simeq [h]_\sim$ by

$$h = (\mathbb{N}^n \times \mathbb{N}^m \xrightarrow{f\times g} |\mathscr{A}| \times |\mathscr{A}| \xrightarrow{(-)\cdot(-)} |\mathscr{A}|),$$

where $(-)\cdot(-)$ is the application operator of a given PCA $\mathscr{A}$. The operator $*$ is 'asynchronous' in the sense that $f\times g$ is involved, and permits 'delay' in the arguments (as streams) with the partial equivalence relation $\sim$.

Lemma 24. *The operator* $(-)*(-)$ *is well defined.*

We adopt the convention that the operation $*$ is left associative, and say that n-LIM($\mathscr{A}$) is *the extension of a* PCA $\mathscr{A}$ *by the non-constructive parallel limits and the asynchronous application*.

This n-LIM($\mathscr{A}$) corresponds to the PCA n-lim($\mathscr{A}$). We can define the counterpart $a-\mathrm{LIM}(\mathscr{A})$ of the PCA a-lim($\mathscr{A}$) in the same way. For any $f:\mathbb{N}^k \rightharpoonup \mathscr{A}$, an element $[f]_\sim$ of n-LIM($\mathscr{A}$) belongs to $a-\mathrm{LIM}(\mathscr{A})$ iff for some $a\in\mathscr{A}$ and for all natural numbers $n_1,\ldots,n_k$ $f(n_1,\ldots,n_k)\simeq a\cdot\langle\overline{n_1},\ldots,\overline{n_k}\rangle$. Then $a-\mathrm{LIM}(\mathscr{A})$ becomes a quotient structure of a partial equivalence relation over $\mathscr{A}$.

Remark 25. Below is explanation of the application operator with the vocabulary of the concurrency theory. Let f,g,h be as in Definition 23.

(1) f and g are processes having, at most n and m independent clocks, respectively. For all time slices[3] except for finite numbers, we can observe $\mathscr{A}$-elements from each f and g.

(2) h is a process having the clocks of both f and g.
If the observation of f at a time slice $\vec{x}\in\mathbb{N}^n$ is $f(\vec{x})$, and the observation of g at another time slice $\vec{y}\in\mathbb{N}^n$, then the observation of h at the time slice $\vec{x},\vec{y}$ is $a\cdot b$.

Lemma 26. *In the structure* n-LIM($\mathscr{A}$), *we have*

$$[k]_\sim * x * y = x.$$
$$[s]_\sim * x * y * z \simeq x * z * (y * z) \quad \textit{if } z = [h]_\sim \textit{ for some } h\in\mathscr{A}.$$

Proof. Let $x=[f]_\sim, y=[g]_\sim$, and the arities of f,g be n,m respectively. Then $[k]_\sim * x * y$ is $[p]_\sim$ such that $p = f\circ(\pi_1^{n+m}\times\cdots\times\pi_n^{n+m}) \sim f$ by the 'exchange'-'weakening' rule. Therefore, $[k]_\sim * x * y = x$.

$[s]_\sim * x * y * z$ is $[u]_\sim$ with u being naturally an $(n+m)$-ary partial function, while $x*z*(y*z)$ is $[v]_\sim$ with v being the same arity as u. We can prove the statement similarly. □

[3] $(t_1,\ldots,t_n)$ where each t_i is the value of the clock.

Definition 27. For every polynomial $t[x]$ over the structure n-LIM($\mathscr{A}$), define the polynomial $\lambda x.t[x]$ as follows. Let x not occur in u.

$$\begin{aligned}
\lambda x.a &\simeq [k]_\sim * a,\\
\lambda x.x &\simeq [\lambda x.x]_\sim,\\
\lambda x.u * x &\simeq u,\\
\lambda x.t[x] * u &\simeq [\lambda xyz.xzy]_\sim * (\lambda x.t[x]) * u,\\
\lambda x.u * t[x] &\simeq [\lambda xyz.x(yz)]_\sim * u * (\lambda x.t[x]),\\
\lambda x.t[x] * t'[x] &\simeq [s]_\sim * (\lambda x.t[x]) * (\lambda x.t'[x]).
\end{aligned}$$

Lemma 28. *Let $t[x]$ be a polynomial over the structure* n-LIM($\mathscr{A}$) *and a be an element of* n-LIM($\mathscr{A}$). *If x occurs in $t[x]$ at most once, or if $a=[h]_\sim$ for some $h\in\mathscr{A}$, then*

$$(\lambda x.t[x])a \simeq t[a].$$

The *Church numeral* $\bar{t}$ below is defined with the λ-abstraction specified by Definition 27.

Lemma 29. *The set* n-LIM($\mathscr{A}$) *has a pairing* **cons**, **car**, **cdr**; *and* **nth** *such that* $\mathbf{nth} * \langle x_0,\dots,x_n\rangle * \bar{t} \simeq x_t$.

Proof. The pairing [2, Definition 6.2.4] is allowed by Lemma 28. Let $\mathbf{nth} \simeq \lambda xy.\mathbf{car} * (y * \mathbf{cdr} * x)$. □

5.3. The interpretation

Convention 1. *In every $\lambda\mu$-term M, every bound μ-variable is distinct*[4] *and different from any free μ-variables.*

Definition 30. Let $\{\alpha_0,\alpha_1,\dots,\alpha_k\}$ be the set of μ-variables in a type-free $\lambda\mu$-term M, then a set $\{t_{\alpha_0},t_{\alpha_1},\dots,t_{\alpha_k}\}$ of numerical variables is denoted by M^p.

We define the partial function $M^g(t_{\alpha_1},\dots,t_{\alpha_k})$ returning at most one $\mathscr{A}$-element, when the values of M^p are determined:

$$\begin{aligned}
x^g &\equiv x,\\
(MN)^g &\equiv M^gN^g,\\
(\lambda x.M)^g &\equiv \lambda x.M^g,\\
([\alpha]M)^g &\equiv M^g(\mathbf{nth}\ \alpha\ \bar{0})\dots(\mathbf{nth}\ \alpha\ \overline{t_\alpha}),\\
(\mu\alpha.M)^g &\equiv \lambda\alpha_0\dots\alpha_{t_\alpha}.M^g[\alpha := \langle\alpha_0,\dots,\alpha_{t_\alpha}\rangle].
\end{aligned}$$

Let $\mathscr{A}$ be any model of λ-calculus. The interpretation $[\![M]\!]$ of M in n-LIM($\mathscr{A}$) is defined as $[M^g]_\sim$.

[4] That is, $\alpha\neq\beta$, if M is $\cdots(\mu\alpha.\cdots\alpha\cdots)\cdots(\mu\beta.\cdots\beta\cdots)\cdots$.

We prove that the above interpretation works well for the *affine* type-free $\lambda\mu$-calculus. By word "affine", we mean that each bound variable as well as each bound μ-variable occurs at most once inside the scope. Being affine is required because of Lemma 28.

Lemma 31. *For all terms P, Q and for all term-variable x, we have*

$$(P[x := Q])^g \equiv P^g[x := Q^g]. \tag{14}$$

Proof. The proof proceeds by induction on P. We abbreviate $[x := Q]$ as θ, and $[x := Q^g]$ as θ^g. Then the left-hand side of (14) is $(P\theta)^g$ and the right-hand side is $P^g\theta^g$.

Case 1: P is $[\alpha]M$.

$$\begin{aligned}
&\text{The left-hand side of (14)}\\
&\equiv ([\alpha](M\theta))^g\\
&\equiv (M\theta)^g\ (\mathbf{nth}\ \alpha\ \bar{0})\cdots(\mathbf{nth}\ \alpha\ \overline{t_\alpha})\\
&\equiv (M^g\ (\mathbf{nth}\ \alpha\ \bar{0})\cdots(\mathbf{nth}\ \alpha\ \overline{t_\alpha}))\theta^g\\
&\quad \text{by induction hypothesis } (M\theta)^g \equiv M^g\theta^g\\
&\equiv \text{the right-hand side of (14).}
\end{aligned}$$

Case 2: P is $\mu\beta.M$.

$$\begin{aligned}
&\text{The left-hand side of(14)}\\
&\equiv (\mu\beta.(M\theta))^g\\
&\equiv \lambda\beta_0\ldots\beta_{t_\beta}.(M\theta)^g\ [\beta := \langle\beta_0,\ldots,\beta_{t_\beta}\rangle]\\
&\equiv \lambda\beta_0\ldots\beta_{t_\beta}.M^g\theta^g\ [\beta := \langle\beta_0,\ldots,\beta_{t_\beta}\rangle]\quad \text{by induction hypothesis.}
\end{aligned}$$

Because we can assume that β is not in Q without loss of generality, the last is $(\lambda\beta_0\ldots\beta_{t_\beta}.M^g\ [\beta := \langle\beta_0,\ldots,\beta_{t_\beta}\rangle])\ \theta^g$, which is the right-hand side of (14).

When P is of the other forms, then the proof is trivial. □

Lemma 32. *For all terms P and Q and for all term-variable x, we have*

$$(P[[\alpha] := (\ )x_0\ldots x_{t_\alpha}])^g \equiv P^g[\alpha := \langle x_0,\ldots,x_{t_\alpha}\rangle]. \tag{15}$$

Proof. Again the proof proceeds by induction on P. Below, the mixed substitution in the left-hand side of (15) is abbreviated as ϑ, and the substitution in the right-hand side of (15) as θ. So, the left-hand side is now $(P\vartheta)^g$ and the right-hand side is $P^g\theta$. We assume that the μ-variable α differs from the μ-variable β.

Case 1: P is $[\beta]M$.

$$\begin{aligned}
&\text{The left-hand side of (15)}\\
&\equiv ([\beta](M\vartheta))^g
\end{aligned}$$

$\equiv (M\vartheta)^g\ (\mathbf{nth}\ \beta\ \bar{0})\cdots(\mathbf{nth}\ \beta\ \overline{t_\beta})$
$\equiv (M^g\ (\mathbf{nth}\ \beta\ \bar{0})\cdots(\mathbf{nth}\ \beta\ \overline{t_\beta}))\theta$ by induction hypothesis
$\equiv$ the right-hand side of (15).

Case 2: P is $[\alpha]M$.

The left-hand side of (15)
$\equiv (M\vartheta\ x_0 \ldots x_{t_\alpha})^g$
$\equiv M^g\theta x_0 \ldots x_{t_\alpha}$ by induction hypothesis
$\equiv$ the right-hand side of (15) .

Case 3: P is $\mu\beta.M$.

The left-hand side of (15)
$\equiv (\mu\beta.M\vartheta)^g$
$\equiv \lambda\beta_0 \ldots \beta_{t_\beta}.(M\vartheta)^g\ [\beta := \langle\beta_0,\ \ldots\ \beta_{t_\beta}\rangle]$
$\equiv \lambda\beta_0 \ldots \beta_{t_\beta}.M^g\theta\ [\beta := \langle\beta_0,\ \ldots\ \beta_{t_\beta}\rangle]$ by induction hypothesis
$\equiv \lambda\beta_0 \ldots \beta_{t_\beta}.M^g[\beta := \langle\beta_0,\ \ldots\ \beta_{t_\beta}\rangle]\theta$
$\equiv$ the right-hand side of (15).

When P is of the other forms, then the proof is trivial. □

Theorem 33. *If a term-variable x occurs free at most once in a term M, or if no μ-variable occurs in a term N, then $[\![(\lambda x.M)N]\!] \simeq [\![M[x := N]]\!]$.*

Proof. From the assumption x occurs free at most once in M^g, or $[\![N]\!] \simeq [h]_\sim$ for some $h \in \mathscr{A}$. So, by Lemmas 28 and 31, the proof is complete. □

Theorem 34. *We can interpret the* Parigot's S3 rule:

$$[\![\mu\alpha.M]\!] \simeq [\![\lambda x.\mu\alpha.M[\,[\alpha] := [\alpha](( )x)]]\!].$$

Proof. We have only to prove that if $f(t_\alpha, t_{\vec{\beta}})$ is $(\mu\alpha.M)^g$, then $f(t_\alpha+1, t_{\vec{\beta}})$ is $(\lambda x.\mu\alpha.M[\,[\alpha] := [\alpha](( )x)])^g$.

$f(t_\alpha + 1, t_{\vec{\beta}})$
$\equiv \lambda\alpha_0 \ldots \alpha_{t_\alpha+1}.M^g[\alpha := \langle\alpha_0, \ldots, \alpha_{t_\alpha+1}\rangle]$
$\equiv \lambda\alpha_0 \ldots \alpha_{t_\alpha+1}\ (M[\,[\alpha] := ( )\alpha_0 \ldots \alpha_{t_\alpha+1}])^g$ by Lemma 32
$\equiv \lambda x\alpha_0 \ldots \alpha_{t_\alpha}.(M[\,[\alpha] := ( )x\alpha_0 \ldots \alpha_{t_\alpha}])^g$
by renaming bound variables.

The last mixed substitution is the mixed substitution $[\,[\alpha] := [\alpha](( )x)\,]$ followed by the mixed substitution $[\,[\alpha] := ( )\alpha_0 \dots \alpha_{t_\alpha}]$. By Lemma 32,

$$\begin{aligned} f(t_\alpha + 1, t_{\vec{\beta}}) &\equiv \lambda x \alpha_0 \dots \alpha_{t_\alpha}.(M[\,[\alpha] := [\alpha](( )x)])^g\ [\alpha := \langle \alpha_0, \dots, \alpha_{t_\alpha} \rangle] \\ &\equiv (\lambda x \mu \alpha.M[\,[\alpha] := [\alpha](( )x)])^g. \qquad \square \end{aligned}$$

Based on Lemma 31, we have the following corollary.

Corollary 35. *Let a μ-variable α occur free at most once in a term P, or let no μ-variable occur in a term Q. Then, we can interpret the μ-reduction*:

$$[\![(\mu\alpha.P)Q]\!] \simeq [\![\mu\alpha.P[\,[\alpha] := [\alpha](( )Q)]]\!].$$

Theorem 36 (η_{cont}). *If a μ-variable α is not free in a term M, then we have $(\mu\alpha.[\alpha]M)^g \equiv M^g$, Hence we can interpret the η_{cont}-rule*:

$$[\![\mu\alpha.[\alpha]M]\!] \simeq [\![M]\!].$$

Proof. The translation $(-)^g$ unwinds each occurrence α^i of α to the same sequence of usual variables $\alpha_0, \alpha_1, \dots, \alpha_{t_\alpha}$. $\square$

If we wish to validate the μ-reduction but not the η_{cont}-reduction, we replace $(-)^g$ with another translation $(-)^G$ satisfying the following two conditions: [5]

(1) For the different occurrences $\alpha^1, \alpha^2, \dots$ of any μ-variable α, $(-)^G$ unwinds α^i to $\alpha_0, \alpha_1, \dots, \alpha_{t_{\alpha^i}}$.
(2) For the μ-reduction with the following graphical presentation:

$$\begin{array}{c} (\mu\alpha^1.(\cdots([\alpha^2]P)\cdots([\alpha^3]Q)\cdots))N \\ \xrightarrow{\ \ \mu\ \ } \\ \mu\alpha^1.(\cdots([\alpha^2](P'N))\cdots([\alpha^3]Q'N)\cdots), \end{array}$$

we have $t_{\alpha^1} \geqslant t_{\alpha^2}, t_{\alpha^3}, \dots$.

6. Related work

6.1. Classical logic as limit

We aim to bridge the gap between constructive and classical logic through (Nakata–Hayashi) realizability interpretations. In order to do so, we are concerned only with the properties of limit operations which are relevant to PCAs.

[5] These two conditions may be expressed along the line of Remark 25 with the vocabulary of the concurrency theory (like, "$\mu\alpha$ waits the α's that occur inside the scope....."). Because we comply Convention 1, we assign the largest possible number of clocks to a given $\lambda\mu$-term. Renaming of bound μ-variables of a $\lambda\mu$-term M can save the number of clocks which is assigned to M. Renaming of bound μ-variables may correspond to a certain scheduling among the parallel executions.

Different uses of limit in interpreting classical logic are introduced by Berardi [3]. He intuitionistically built a constructive model for the Δ_2^0 maps. His model is a refinement of constructive interpretations for classical reasoning over one-quantifier formulas.

In his model, he used a completion idea which is similar to the topological completion producing $\mathbb{R}$ out of $\mathbb{Q}$. One of his concerns was to clarify a constructive process of computing the limiting value. Based on those processes, he directly interpreted his semi-formal system of Δ_2^0 maps.

The main differences between Berardi's work [3] and our own results are as follows:

(1) In intuitionistically constructing a model of the Δ_2^0 maps, Berardi first collected any total recursive unary function f such that $\lim_t f(t)$ is "convergent". Then he quotiented the collection by the cofinal equality $=^*$. Berardi intuitionistically defined "convergence", which happened to be closely related to the change-of-mind ordering which appeared in Ershov's Boolean hierarchy [1]. Berardi's cofinal equality for unary functions f and g was defined as

$$f =^* g : \Leftrightarrow \forall n \exists m > n. f(m) = g(m).$$

The cofinal equality for converging unary functions is classically equivalent to our eventual equality

$$\xi \sim \zeta : \Leftrightarrow \exists t_0 \forall t > t_0\, \xi(t) = \zeta(t).$$

On the other hand, we employed classical logic to construct a model a-lim($\mathscr{A}$) of the limiting partial recursive functions. We first collect all partial unary functions, representable in $\mathscr{A}$, and then we quotient the set by the eventual equality $\sim$.

(2) When we write $\lim_{t\to\infty} \xi(t,x)$, we think of t as the clock of some guessing function $\xi(t,x)$, which eventually (from some t_0) stabilizes to some limit value. Berardi also considered t as time, but rather preferred to think of t as the finite set $t = \{y_1, \ldots, y_n\}$ where he used y_i's to check the correctness of a guessing function $\xi(y,x)$. He presupposed that there is a total ordering on the range of values of $\xi(t,x)$, as is the case when ξ guesses the minimum witness of a law of excluded middle $\exists t. p(t,x) \vee \forall t. \neg p(t,x)$. The formula can be realized by $\langle 1,0\rangle$ if $p(t,x)$ is false for all t, or $\langle 0,t\rangle$ if $p(t,x)$ is true. Berardi ordered realizers lexicographically and let $\xi(t,x)$ be the minimum realizer in $\{\xi(y,x) \mid y \in t\}$. As t increases (as a set), $\xi(t,x)$ eventually stabilizes, but in the way different from ours. His aim was to represent computations in which the value of guessing function ξ need not to be checked against all natural numbers.

6.2. Approximating realizers

Nakata and Hayashi [13] approximated a realizer of the Σ_2^0-double negation elimination axiom by a recursive sequence of recursive realizers. Their approximation is not a terminating process.

On the other hand, Berardi et al. [4] approximated a Kleene–Kreisel continuous realizer of the negative translation of the choice axiom. Their approximation terminates because of the continuity of the realizer.

Acknowledgements

The author acknowledges Susumu Hayashi, Mariko Yasugi, Stefano Berardi, Kenetsu Fujita, and Irina Kozlova. The author acknowledges an anonymous referee for his improvement of the presentation of the paper.

References

[1] C.J. Ash, J.F. Knight, Computable Structures and the Hyperarithmetical Hierarchy, Elsevier, Amsterdam, 2000.
[2] H.P. Barendregt, The Lambda Calculus, Its Syntax and Semantics, 2nd ed., North-Holland, Amsterdam, 1984.
[3] S. Berardi, Classical logic as limit completion I& II, Torino University, 2001, manuscript.
[4] S. Berardi, M. Bezem, Th. Coquand, On the computational content of the axiom of choice, J. Symbolic Logic 63 (1998) 600–622.
[5] A. Berarducci, M. Dezani-Ciancaglini, Infinite lambda-calculus and types, Theoret. Comput. Sci. 212 (1999) 29–75.
[6] M. Felleisen, D.P. Friedman, Control operators, the SECD machine, and the λ-calculus, in: M. Wirsing (Ed.), Formal Descriptions of Programming Concepts III, North-Holland, Amsterdam, 1986, pp. 193–217.
[7] K. Fujita, On proof terms and embeddings of classical substructural logics, Studia Logica 61 (1998) 199–221.
[8] E.M. Gold, Limiting recursion, J. Symbolic Logic 30 (1965) 28–48.
[9] G. Grätzer, Universal Algebra, D. Van Nostrand, Princeton, NJ, 1968.
[10] J.M.E. Hyland, P.T. Johnstone, A.M. Pitts, Tripos theory, Math. Proc. Cambridge Philos. Soc. 88 (2) (1980) 205–231.
[11] M. Kanazawa, private communication, 2000.
[12] J.R. Kennaway, J.W. Klop, M.R. Sleep, F.J. de Vries, Infinitary lambda calculus, Theoret. Comput. Sci. 175 (1997) 93–125.
[13] M. Nakata, S. Hayashi, A limiting first order realizability interpretation, Scientiae Mathematicae Japonicae 5 (2001) 567–580.
[14] P. Oddifreddi, Classical Recursion Theory, Vols. I, II, North-Holland, Amsterdam, 1989.
[15] M. Parigot, $\lambda\mu$-calculus: an algorithmic interpretation of classical natural deduction, in: Proc. Internat. Conf. on Logic Programming and Automated Reasoning, Lecture Notes in Computer Science, Vol. 624, Springer, Berlin, 1992, pp. 190–201.
[16] Th. Streicher, B. Reus, Classical logic, continuation semantics and abstract machines, J. Funct. Programming 8 (1998) 543–572.
[17] H.R. Strong, Algebraically generalized recursive function theory, IBM J. Res. Development 12 (1968) 465–475.
[18] E.G. Wagner, Uniformly reflexive structures: on the nature of Gödelizations and relative computability, Trans. Amer. Math. Soc. 144 (1969) 1–41.
[19] T. Yamazaki, private communication, 2000.